(m,n)-mer - A Simple and New Statistical Feature for viral Classification

***Amanda Araújo Serrão de Andrade*[1], *Marco Grivet*[2], *Otávio Brustolini*[1], *and Ana Tereza Ribeiro de Vasconcelos*[1]**

[1]National Laboratory for Scientific Computing, Bioinformatics Laboratory (LABINFO), Petrópolis, Rio de Janeiro, Brazil
[2]Pontifícia Universidade Católica do Rio de Janeiro, 22451-900, Brazil

22 novembro 2022

Abstract

The (m,n)-mer is a new statistical feature based upon conditional frequencies (conditional probability density distributions). Here, we present the mnmer function and show a pratical example of classification using mnmer output.

Package

mnmer 0.99.0

Contents

1 Introduction to the (m,n)-mer concept

The (m,n)-mer R package was created to summarize biological data into numerical characteristics, as an alternative for k-mers. It reads a FASTA file and generates a table describing the conditional frequency distribution of the selected (m,n)-mer in the sequences. The feature matrix for classification is created by combining its output with class information.

Since letters are a bit awkward from a mathematical viewpoint, lets univocally associate the digits 0, 1, 2 and 3 to letters A, C, G and T. Any order will do. Hence, each k-mer can be described by a unique base-4 integer number in the range from 0 to $4.k-1$. As an example, consider the 6-mer ACCTGA and the association described above. Then the 6-mer ACCTGA can be represented as the base-4 number 011320, which corresponds to the number 376 in the decimal notation.

If we order all the k-mers according to these numbers, we say the k-mers are "lexicographically" ordered and we can use the notation s_k^i to identify the k-mer associated to the decimal number i ranging from 0 to $4.k-1$. Following the above example, we can say that $s_6^{376} = ACCTGA$. Let's now consider a particular genome extracted from a particular organism and assume that we count the occurrence of all distinct k-mers present in this genome. We will denote by c_k^i the count corresponding to the i-th k-mer. Then we can create a real vector of size $4.k$ as illustrated in the Figure 1 below:

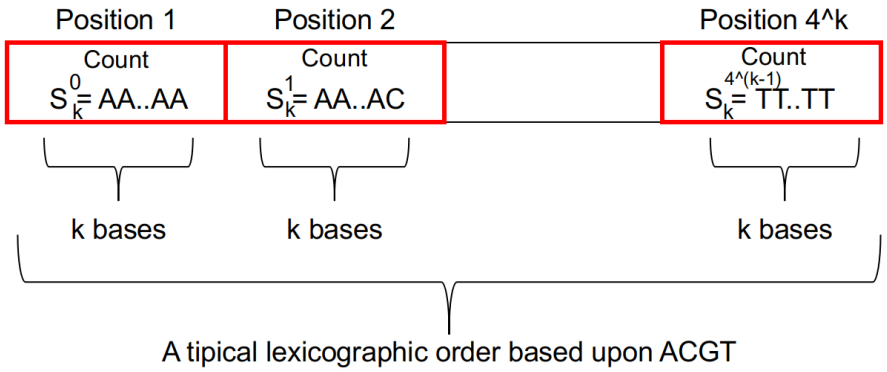


Figure 1: k-mer organization for k-mer and (m,n)-mer statistical distributions evaluation.

Since this counting is generally made for several organisms with different sizes, for the sake of comparison it is more convenient express this count in relative terms. Similarly, we will denote by as f_k^i the relative frequency of k-mer s_k^i which is computed as the division $f_k^i = \frac{c_k^i}{N_k}$ where N_k is the total number of k-mers counted in this organism, that is, $N_k = \sum_{i=0}^{4^k-1} c_k^i$.

Let's define a vector $\underline{f}_k = (f_k^0, f_k^1, ..., f_k^{4^k-1})$ as the formal descriptor of the particular organism as far as k-mer is concerned.

Please notice that each element of vector $\underline{f}_k$ is nonnegative and their sum is 1, which allow us to interpret this vector as a "probability density distribution" according to statistical parlance.

In the above example, consider the 6-mer ACCTGA and m and n respectively assuming the values 4 and 2. Hence we have $S_6 = ACCTGA$, $S_4^- = ACCT$ and $S_2^+ = GA$. Although we have used the superscripts – and $+$ to respectively indicate the left and right part of the k-mer S_k, they are also an m-mer and an n-mer on its own. We propose the replacement of the unconditional frequency f_k^i by the conditional frequency $f_{m,n}^i$ which represents the relative frequency of the n-mer S_n^+ conditioned to the fact that the set of m bases that

precedes it is S_m^-. The vector so defined will be conveniently renamed $\underline{f}_{m,n}$ in order to be more explicit. By defining u/v as the result of the integer division of u by v then, based upon the classical result of conditional probabilities, we have $\underline{f}_{m,n} = (f_{m,n}^0, f_{m,n}^1, ..., f_{m,n}^{4^k-1})$ where $f_{m,n}^i = \frac{f_k^{i]}}{f_m^{i/4^n}}$.

Again in our example, f_6^{376}, the relative frequency of 6-mer ACCTGA must be divided by f_4^{23}, which is the relative frequency of $S_4^{23} = ACCT$.

It can be easily seen that the sum all elements of this new vector is no longer 1 but $4m$, because it is the concatenation of $4m$ conditional frequency distributions associated to each one of the possible m-mers. This fact has no impact whatsoever in our current discussion but, in order to keep it as a probability density distribution as well, we normalize it by dividing it by $4m$. Figure 2 shows an comparison of k-mers and (m,n)-mers obtained from the same nucleotide sequence.

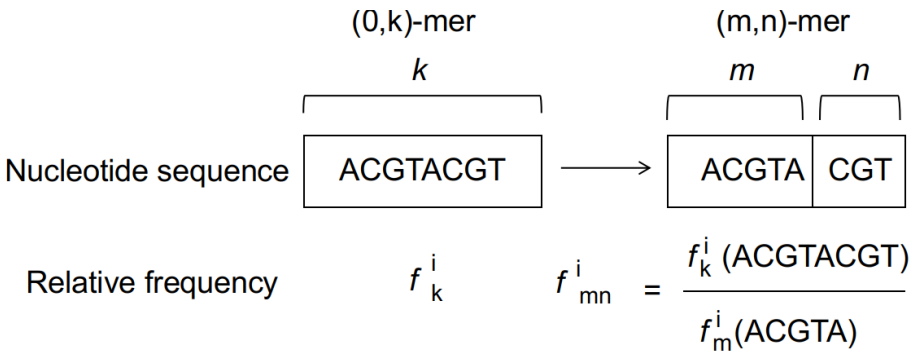


Figure 2: Comparing k-mer to mn-mer relative frequency.

According to Figure 3 below, the k-mers are represented as (0,k) and the (m,n)-mers as (m,n).

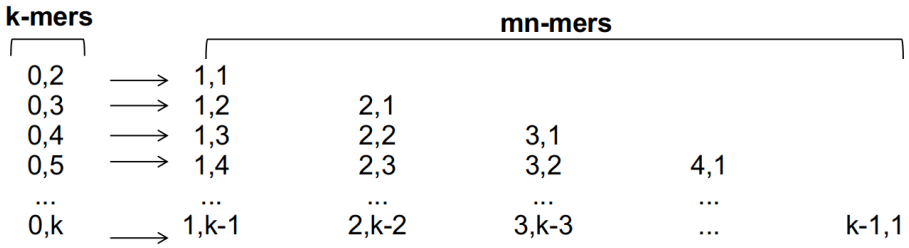


Figure 3: Numeric representation.

The output table (Figure 4) includes the fasta file accession numbers as an ID column, the relative frequency of mn-mers up to $4.k$ columns, and class information.

For more details and performance comparison, please see Andrade et al., 2022 (in press).

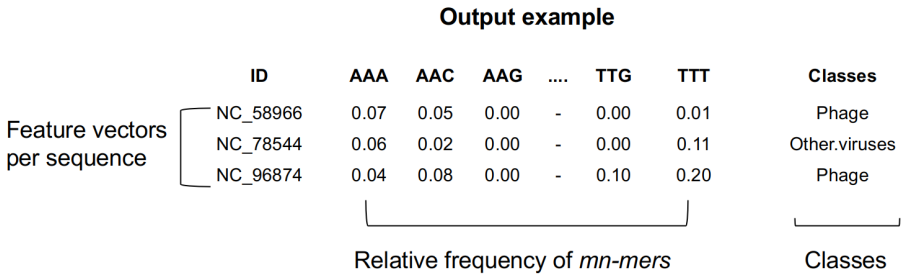


Figure 4: Output example.

2 The (m,n)-mer as an R package

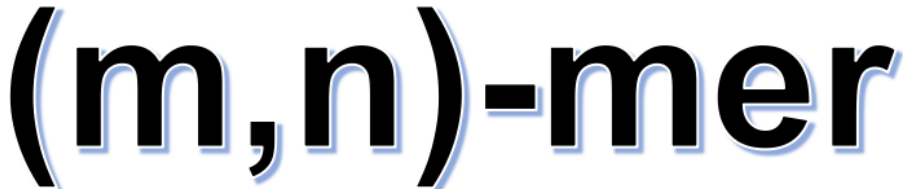


Figure 5: Package logo

The *mnmer* R package was created to summarize biological data into numerical characteristics. It accepts a FASTA file and generates a dataframe that describes the frequency of all (m,n)-mers identified in the input sequences. This output is coupled with class information to create a feature matrix.

2.1 Dependencies

The package only needs *R* 4.0.0 or later.

2.2 Instalation

The user should install the package from the GitHub repository. It can be done by using the *devtools* package.

```
library(devtools)
install_github("labinfo-lncc/mnmer", ref="main")
```

2.3 The mnmer function

The main function of this package is the `mnmer` function. It creates dataframes with the conditional probability. By invoking the function `cmmer` from the C++ script, this function can create both k-mers and (m,n)-mers.

The parameters receives:

`file` = file = FASTA file. It could be a multiFASTA. This file can be .gz compressed or not.

`k` = Value of k for k-mer generation. Needs to be different from zero.

`m` = Value of m for (m,n)-mer generation in the format of (m, k-m). In case of k-mer generation, m should be zero as (0,k).

As default, all sequences with high content of N + IUPAC bases will be removed from further analysis given the little informative nature of those bases. In that case, the mnmer function prints the following warning:

```
## [1] "Warning: Sequence has a proportion of N + IUPAC bases = 10%"
```

2.4 Pratical example

The mnmer function generates a independent feature matrix that may be used to conduct clustering or classification.

Assume we need to distinguish between viruses that only replicate in mosquito and viruses that only replicate in plants. The corresponding FASTA files can be found in the extdata folder.

After package installation, the user should run:

```
library("mnmer")
dir <-system.file("extdata", package="mnmer")
```

2.4.1 Produce k-mer distributions

For k-mer generation, the parameter k is set to choice, while the parameter m is set to zero. Given that the k-mers have been conditioned to zero bases.

```
mosquito <- mnmer(file.path(dir, "mosquito_vir.fasta"),2,0)
plant <- mnmer(file.path(dir, "plant_vir.fasta"),2,0)
```

2.4.2 Produce (m,n)-mer distributions

The user specifies the k and m parameters for (mn)-mer generation.

For example, k = 2 and m = 1 produce the (1,1)-mer, in which one base is conditioned on the frequency of the base before it. Bases other than A, C, T, and G were disregarded.

```
mosquito <- mnmer(file.path(dir, "mosquito_vir.fasta"),2,1)
plant <- mnmer(file.path(dir, "plant_vir.fasta"),2,1)
```

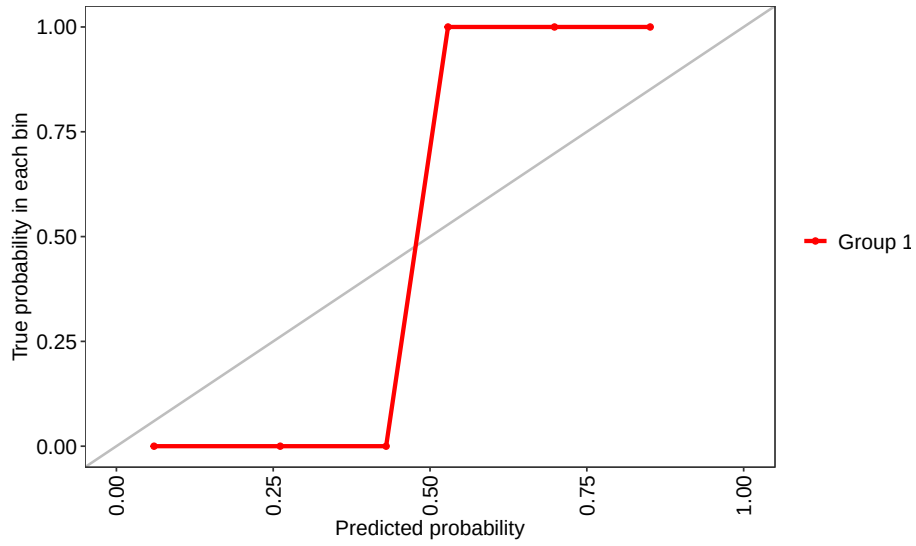
Here, we utilize the (1,1)-mer feature matrices generated by the mnmer to run an classification using Caret and MLeval.

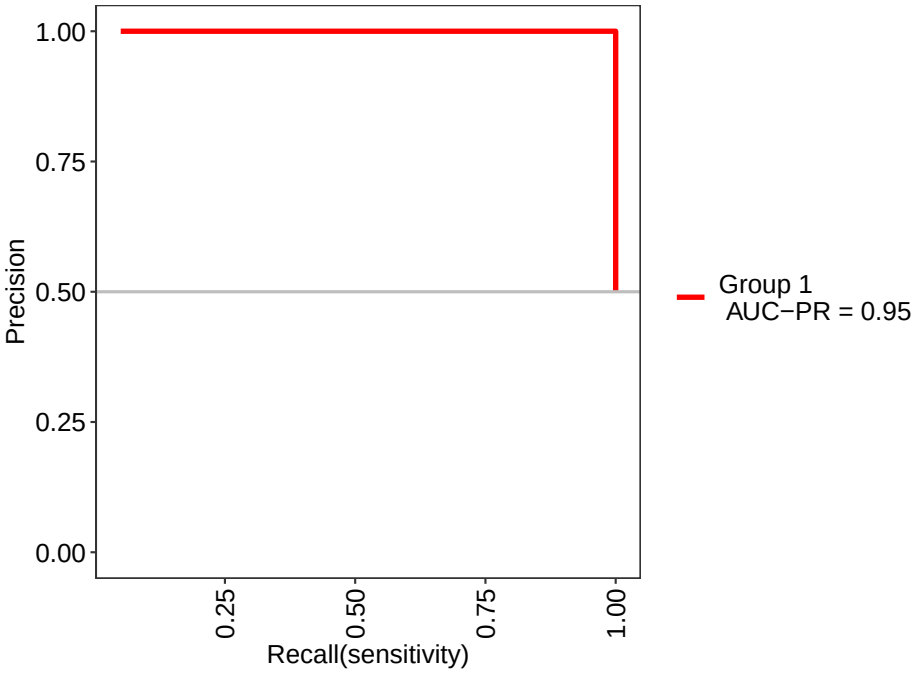
Caret (https://topepo.github.io/caret/) is a library of functions for building predictive models from R systems. We utilized the createDataPartition, trainControl, and train functions in this example. The createDataPartition method splits the feature matrix and creates the train and test datasets. The trainControl function generates parameters that further regulate how the train function creates models.

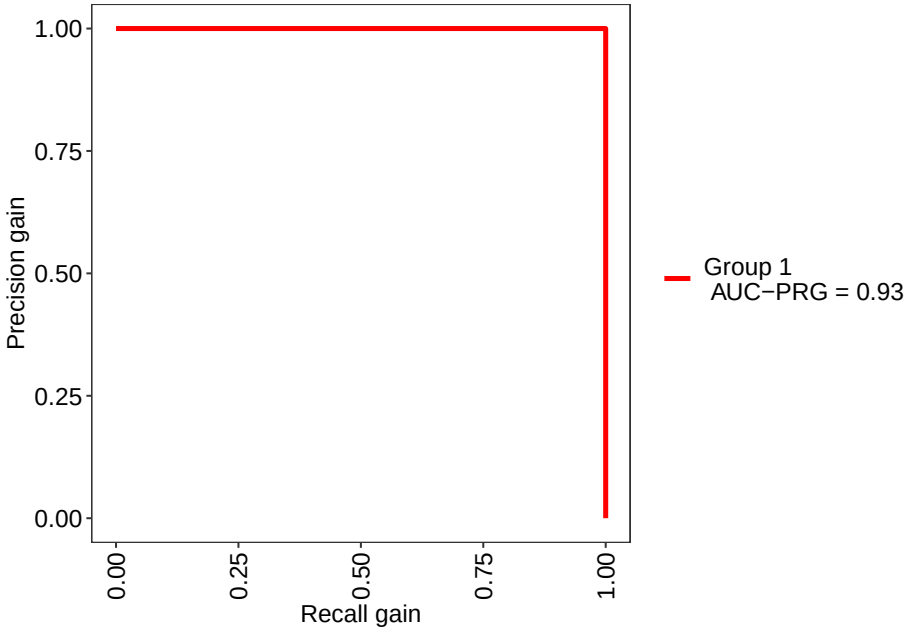
MLeval (https://cran.r-project.org/web/packages/MLeval/index.html) is used to plot performance metrics.

To execute the example, enter the following code:

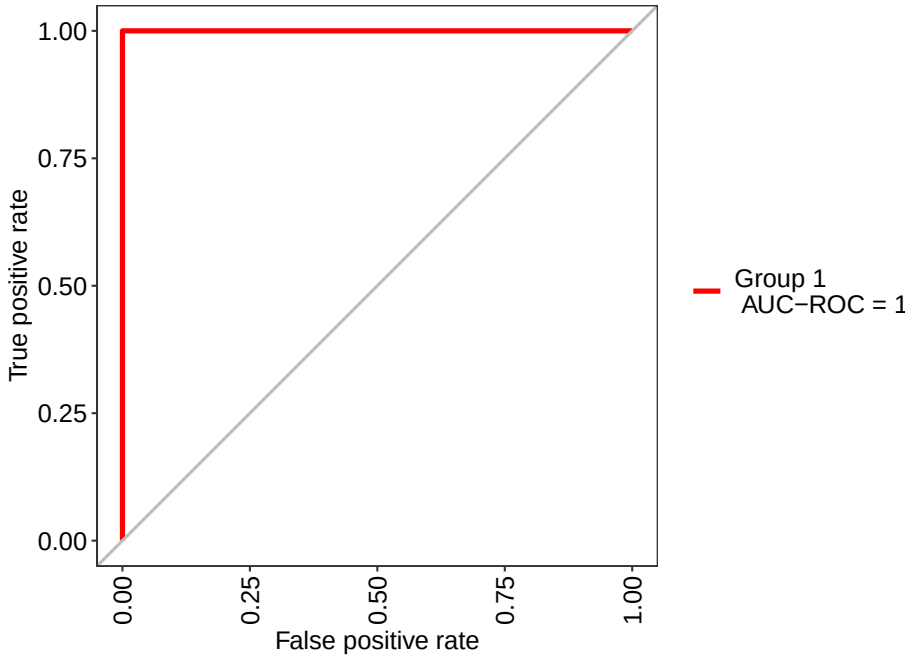
```
library(caret)
## Carregando pacotes exigidos: ggplot2
## Carregando pacotes exigidos: lattice
# Add class information
classes <- replicate(nrow(mosquito), "mosquito.vir")
featureMatrix_mosquito <- cbind(mosquito,classes)
classes <- replicate(nrow(plant), "plant.vir")
featureMatrix_plant <- cbind(plant,classes)
featureMatrix <- rbind(featureMatrix_mosquito, featureMatrix_plant)
featureMatrix <- subset(featureMatrix, select = -c(seqid))

# Machine Learning
train_index <- caret::createDataPartition(featureMatrix$classes, p=0.8, list=FALSE)
train <- featureMatrix[train_index, ]
test <- featureMatrix[-train_index, ]
control <- caret::trainControl(method="cv",
                               summaryFunction=twoClassSummary,
                               classProbs=TRUE,
                               savePredictions = TRUE)
roc <- caret::train(classes ~ .,
                    data=train,
                    method="rf",
                    preProc=c("center"),
                    trControl=control)
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
res <- MLeval::evalm(roc) # Make the ROC plot
## ***MLeval: Machine Learning Model Evaluation***
## Input: caret train function object
## Not averaging probs.
## Group 1 type: cv
## Observations: 40
## Number of groups: 1
## Observations per group: 40
## Positive: plant.vir
## Negative: mosquito.vir
## Group: Group 1
## Positive: 20
## Negative: 20
## ***Performance Metrics***
```


1.00
0.75
0.50
0.25
0.00
True probability in each bin
0.00
0.25
0.50
0.75
1.00
Predicted probability
Group 1


1.00
0.75
0.50
0.25
0.00
Precision
0.25
0.50
0.75
1.00
Recall(sensitivity)
Group 1
AUC−PR = 0.95




```
## Group 1 Optimal Informedness = 1
## Group 1 AUC-ROC = 1
```



3 SessionInfo

```
## R version 4.2.1 (2022-06-23)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.5 LTS
```

```
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/liblapack.so.3
##
## locale:
##  [1] LC_CTYPE=pt_BR.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=pt_BR.UTF-8        LC_COLLATE=pt_BR.UTF-8
##  [5] LC_MONETARY=pt_BR.UTF-8    LC_MESSAGES=pt_BR.UTF-8
##  [7] LC_PAPER=pt_BR.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=pt_BR.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] caret_6.0-93     lattice_0.20-45  ggplot2_3.4.0    mnmer_0.99.0
## [5] BiocStyle_2.26.0
##
## loaded via a namespace (and not attached):
##  [1] splines_4.2.1          foreach_1.5.2          prodlim_2019.11.13
##  [4] assertthat_0.2.1       BiocManager_1.30.19    stats4_4.2.1
##  [7] GenomeInfoDbData_1.2.9 yaml_2.3.6             globals_0.16.1
## [10] ipred_0.9-13           pillar_1.8.1           glue_1.6.2
## [13] pROC_1.18.0            digest_0.6.30          XVector_0.38.0
## [16] randomForest_4.7-1.1   hardhat_1.2.0          colorspace_2.0-3
## [19] recipes_1.0.2          htmltools_0.5.3        Matrix_1.5-1
## [22] plyr_1.8.7             timeDate_4021.106      pkgconfig_2.0.3
## [25] listenv_0.8.0          bookdown_0.29          zlibbioc_1.44.0
## [28] purrr_0.3.5            scales_1.2.1           gower_1.0.0
## [31] lava_1.7.0             tibble_3.1.8           farver_2.1.1
## [34] generics_0.1.3         IRanges_2.32.0         withr_2.5.0
## [37] nnet_7.3-18            BiocGenerics_0.44.0    cli_3.4.1
## [40] survival_3.4-0         magrittr_2.0.3         crayon_1.5.2
## [43] evaluate_0.17          fansi_1.0.3            future_1.28.0
## [46] parallelly_1.32.1      nlme_3.1-160           MASS_7.3-58.1
## [49] class_7.3-20           tools_4.2.1            data.table_1.14.4
## [52] lifecycle_1.0.3        stringr_1.4.1          S4Vectors_0.36.0
## [55] munsell_0.5.0          Biostrings_2.66.0      compiler_4.2.1
## [58] GenomeInfoDb_1.34.1    rlang_1.0.6            grid_4.2.1
## [61] RCurl_1.98-1.9         iterators_1.0.14       rstudioapi_0.14
## [64] labeling_0.4.2         bitops_1.0-7           rmarkdown_2.17
## [67] MLeval_0.3             gtable_0.3.1           ModelMetrics_1.2.2.2
## [70] codetools_0.2-18       DBI_1.1.3              reshape2_1.4.4
## [73] R6_2.5.1               lubridate_1.8.0        knitr_1.40
## [76] dplyr_1.0.10           fastmap_1.1.0          future.apply_1.9.1
## [79] utf8_1.2.2             stringi_1.7.8          parallel_4.2.1
## [82] Rcpp_1.0.9             vctrs_0.5.0            rpart_4.1.16
## [85] tidyselect_1.2.0       xfun_0.34
```